OPENGL BASED 3D FIRST PERSON SHOOTING GAMES – DESIGN CONCERNS

W. G. C. W Kumara.[1], K Wattanachote.[2]

[1] MINE Lab, Department of Computer Science and Information Engineering, National Central University, Taiwan.
Email: **chinthakawk@gmail.com**
[2] MINE Lab, Department of Computer Science and Information Engineering, National Central University, Taiwan.
Email: kanoksak.wattanachote@gmail.com

ABSTRACT

3D video games are getting popular in the world with the availability of advanced graphic cards, high processing power computers, high speed internet and smart sensing devices ranging from general mouse to Microsoft Kinect. OpenGL is a popular graphics processing framework and it is being used by many famous 3D video game design software as the back end framework. In this paper we present our experience with OpenGL based C++ implementation of a 3D first person shooting game. 3D environment building, navigating, character animation, lighting, sound and shooting is described. Specially OpenGL based concepts are discussed for clear understanding of the concepts.

***Key words*:** 3D video games, animation, first person shooting, OpenGL

1. INTRODUCTION

We can see that use of 3D games online or offline is intensively increasing even though true statistics are not present. Especially there are many research effects are going on in different fields of 2D/3D gaming as graphics, sounds, virtual reality, game strategies etc. Since students spend more time on games there are new research areas are exploiting to use that time effectively like to help them learn some subject matters or language concepts etc.

OpenGL is the premier environment for developing portable interactive 2D and 3D graphics applications, since its introduction in 1992 and has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms [1]. There are many game development software available in use [2] while many of them are based on the OpenGL platform [3]. Unity[1] and XNA[2] are two of famous software within 3D games development community for Windows.

Main objective of this paper is to explore and summarize the 3D game design concerns used in now a day. We present our experience with development of a 3D game based on OpenGL on Windows 7 platform as well.

2. DESIGN AND IMPLEMENTATION

Here we describe the design concerns and implementation of our game using OpenGL whenever necessary.

(a) Game interface
We first designed the interface of the game as in Figure 1.

We used *viewport* function in OpenGL to divide the screen based on our requirement. Following example code shows how to divide the screen.

```
glEnable(GL_SCISSOR_TEST);
glViewport(x,y,width,height);
glScissor(x,y,width,height);
glDisable(GL_SCISSOR_TEST);
```

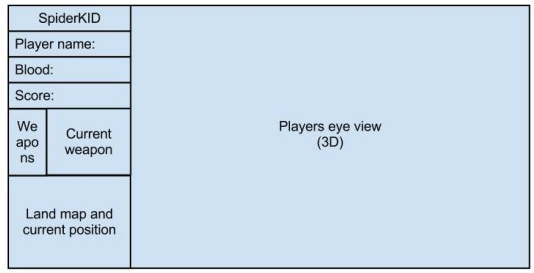
(b) Lettering
OpenGL supports stroke and raster text formats. We used the following function provided in the GLUT library[3].

```
glPushMatrix();
glColor3f(1,0,0);
renderBitmapString(-15,0,-
10,GLUT_BITMAP_8_BY_13,"SpiderKID");
glPopMatrix();
```

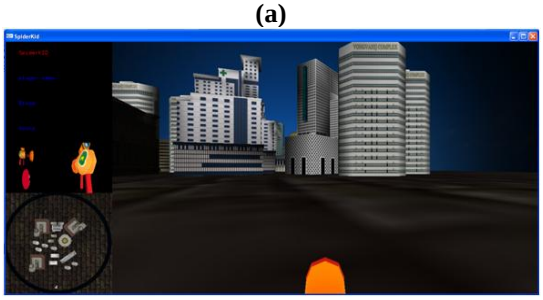
[1] http://unity3d.com/

[2] http://msdn.microsoft.com/en-us/centrum-xna.aspx

[3] http://www.opengl.org/resources/libraries/glut/



(a)



(b)

Figure 1: Main interface of the game (a) concept (b) implementation

(c) Creating models

Creating models in OpenGL is straight forward. There are several coordinate spaces available in OpenGL as object, world, camera, screen coordinates etc. 3D coordinates of the object should be spcified first w.r.t. object coordinates then can be tranferred to screen coordinates at last through several steps based on our requirement. Since we have several complex buildings in our design 3DMax was used to design the building first and then imported to OpenGL environment. A *.obj* file is used to provide all the 3D coordinates and *.mtl* file is used to provide the relevant material information of the surfaces as texture information.

(d) Camera setting

OpenGL *gluLookAt* function is used to specify the viewing direction as follows. First three parameters provide the x,y,z coodinates of the viewer location and follows by the coodinates of the looking at location and the look up vector located at the viewer.

```
gluLookAt(viewer[0],viewer[1],viewer[
2],viewer[0]+cos(rot),lookat[1]+sin(v
rot),viewer[2]+sin(rot),0,1,0);
```

glOrtho projection method is used in the map view window to have a orthogonal view.

```
glMatrixMode(GL_PROJECTION);
glOrtho(-0.7, 0.7, -0.7, 0.7, 0.1,
0.5);
glMatrixMode(GL_MODELVIEW);
```

glPerspective projection is used in the main window to have a perspective view.

```
glMatrixMode(GL_PROJECTION);
gluPerspective(60,(GLfloat)width/(GLf
loat)(height),0.000001, 1000);
glMatrixMode(GL_MODELVIEW);
```

(e) Navigating through the environment

Navigating through the environment is necessary in the 3D games. Since variables are used in the gluLookAt function upon pressing of arrow keys we navigate through the scene. Left and right arrow keys are used to turn left and right while up and down arrows are used to go forward and backward.

```
void arrows(GLint key, GLint x, GLint
y)
{
switch (key)
{
case GLUT_KEY_LEFT:
   rot-=ROTATE_KEY_SENSE; break;
case GLUT_KEY_RIGHT:
    rot+=ROTATE_KEY_SENSE; break;
case GLUT_KEY_UP:
    move(WALK_KEY_SENSE); break;
case GLUT_KEY_DOWN:
    move(-
WALK_KEY_REVERSE_SENSE);break;
}
glutPostRedisplay();
}

void move(GLfloat amt)
{
viewer[0]+=cos(rot)*amt;
viewer[2]+=sin(rot)*amt;
}
```

Figure 2: Map view

A small red color triangle is used to show the current location in the map view while changing the direction based on viewer direction as shown in the following example.

```
glPushMatrix();
glTranslatef(viewer[0], 0,
viewer[2]);
draw_triangle();
glPopMatrix();
```

(f) Lighting

Type of the light, color of the rays and material properties etc should be considered when design lighting of the game. We used following functions to change lighting of the environment.

```
GLfloat
Ambient[]={0.1f,0.1f,0.1f,1.0f};
GLfloat
Diffuse[]={1.0f,1.0f,1.0f,1.0f};
GLfloat
Position[]={10.0f,60.0f,10.0f,1.0f};
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightfv(GL_LIGHT0, GL_AMBIENT,
Ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE,
Diffuse);
glDisable(GL_LIGHTING);
```

(g) Fog effect

Fog effect in the environment helps to create a more realistic scenary in the 3D game. This is how we used fog effect in the game.

```
GLfloat density = 0.3;
GLfloat fogColor[4] = {1.0, 1.0, 1.0,
1.0};
glEnable(GL_FOG);
glFogi(GL_FOG_MODE, GL_EXP);
glFogfv(GL_FOG_COLOR, fogColor);
glFogf(GL_FOG_DENSITY, density);
glHint(GL_FOG_HINT, GL_NICEST);
glDisable(GL_FOG);
```

(h) Enemies

There are different types of enemies are used in the games. Since this game was designed focusing kids, only one fancy character is used as the enemy (Figure 3). Again 3DMax was used to design the character and imported to OpenGL environment.

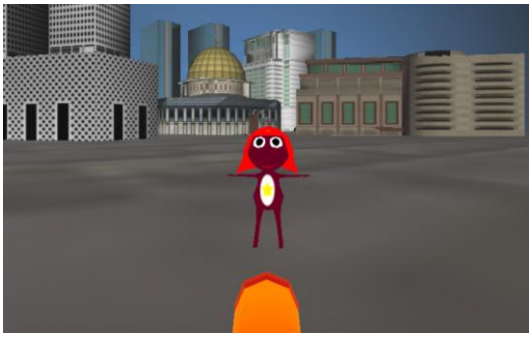


Figure 3: Enemy soldier in the game

(i) Collisions

Collisions are really required in the 3D games as the user should not be allowed to go through the rigid objects like buildings and enemies. There are several ways of handling collisions in 3D games and we used a simple method called *bounding boxes*. A virtual box is created bounding minimum and maximum x and z coordinates (base). Then whenever the player makes a move we check against those virtual bounding boxes. There are several limitations of this method as scalability with high number of objects and less accuracy.

(j) Weapons

There should be several weapon types available in the game. We implemented only two weapons as a shot gun and a bomb. User is allowed to select the weapon by pressing *g* anb *b* keys consecutively for shotgun and bomb. Available weapons and the current weapon is shown in a portion of the user window as in Figure 4. And it is required to show a part of the weapon in the main window as in Figure 1 as seen by the player.

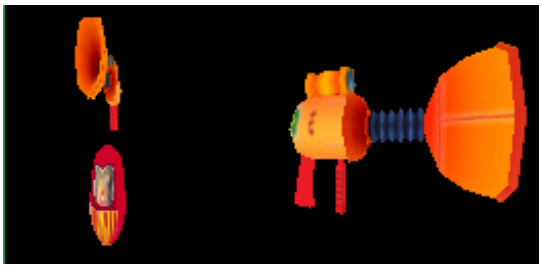


Figure 4: Weapons window

(k) Shooting

Shooting enemies is again very important in the game and game scoring mechanism can be built based on that. A linked list can be used to store the viewer location, target location and the bullet number for each bullet since bullet direction is not changing after released and since all the bullets should be viewed in the scene till it hits the target or dissaper from our visibility. Since focus of this paper is to describe the OpenGL experince and since a shot gun is used implementation of the complex shooting is not discussed here. Same collision detection concepts disscessed in (i) are used to detect collision of the bullets with the target or other objects. *s* key press is used to release the bullet.

(l) Sound

Sound is also vital part of 3D games and one example sound use in the game is given below.

```
PlaySound(TEXT("BOMB.wav"),NULL,SND_A
SYNC|SND_FILENAME);
```

(m) Animations
Handling animations needs special care. As given in Figure 5 a relationship between the components of the object should be clearly defined and proper data structure should be used to implement [4], [6]. Due to the limitations of the paper animations is not discussed in detail here.

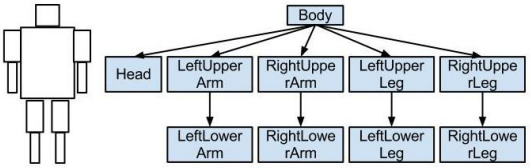


Figure 5: Handling animations [4]

(n) Interactions
Only keyboard interactions are used in the design as up, down, left and right arrow keys and several other character keys. Use of mouse also straigh forward task even though it is not discussed here. Special sensor devices like Microsoft Kinect is also very attractive to be used as an interface to the game.

3. PERFORMANCE

Even though several buildings, lighting effects, fog effects, continues detection of collisions etc. are used in the implementation, we can see that the utilization of the computer resources in Windows 8 when using the game like processor and memory is still low giving room to further complex implementations (Figure 6).

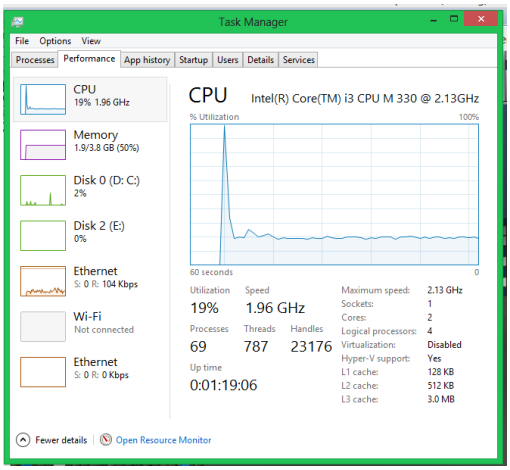


Figure 6: Performance of the implementation

4. CONCLUSIONS

3D games are creating good opportunities for the developers, researches, manufactures and companies etc. in computer science related domain. Design and implementation related concerns faced during design and implementation of an OpenGL based 3D FPS game is discussed here. We hope to present animation, shooting, collision related concepts in future works. Implementation on open source platform like Linux will also be considered in future.

5. REFERENCES

[1] http://www.opengl.org/about/

[2] http://en.wikipedia.org/wiki/Category:Video_game_development_software

[3] http://www.opengl.org/products/

[4] E. Angel and D. Shreiner, "*Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL"*, 6th Ed. Addison-Wesley, 2011.

[5] Q. Lin, Z. Zhao, D. Xu, R. Wang. "*Design and implementation of an openGL based 3D first person shooting game"*. In Transactions on edutainment V, Zhigeng P. Adrian, D. Cheok, W. Müller, X. Yang (Eds.), S.Verlag, B. Heidelberg, pp. 50-61, 2011.

[6] T K. Shih, "*Lecture notes - Interactive Computer Graphics in Game Design"*, Fall National Central University, Taiwan, 2012.